

“Express Mail” Mailing Label No. EL917901459US

PATENT APPLICATION
ATTORNEY DOCKET NO. SUN-P6120-RSH

5

10 **METHOD AND APPARATUS FOR CLASS
INITIALIZATION BARRIERS AND ACCESS TO
CLASS VARIABLES IN MULTITASKING
VIRTUAL MACHINES**

15

Inventor: Laurent P. Daynes and Grzegorz J. Czajkowski

Related Application

20

[0001] This application hereby claims priority under 35 U.S.C. §119 to a Provisional Patent Application entitled, “The Design of MVM—a Multitasking Virtual Machine,” filed March 15, 2001 by inventors Grzegorz J. Czajkowski and Laurent P. Daynes (Application No. 60/276,049).

25

[0002] The subject matter of this application is related to the subject matter in a co-pending non-provisional application by the same inventors as the instant application and filed on the same day as the instant application entitled, "Method and Apparatus to Facilitate Sharing Instruction Code in a Multitasking Virtual Machine," having serial number TO BE ASSIGNED, and filing date TO BE ASSIGNED (Attorney Docket No. SUN-P6119-RSH).

BACKGROUND

Field of the Invention

5 [0003] The present invention relates to computer instruction code. More specifically, the present invention relates to a method and an apparatus for a class initialization barrier in a multitasking virtual machine.

Related Art

10 [0004] Computer programs written in languages such as JAVA™ are compiled into a platform-independent code, which is executed on a virtual machine, such as a JAVA VIRTUAL MACHINE (JVM). A program that has been compiled into a platform-independent code has the advantage that it can execute on a virtual machine regardless of the underlying central processing unit and native code. The terms JAVA, JVM, and JAVA VIRTUAL MACHINE are
15 trademarks or registered trademarks of SUN Microsystems, Inc. of Palo Alto, California.

 [0005] A virtual machine typically includes an interpreter, which interprets the platform-independent code into native code to perform the desired operations. Interpreting the platform-independent code is an inherently slow operation.
20 Therefore, many virtual machine implementations also include a dynamic compiler, which can compile at runtime the platform-independent code into the native code of the machine being used to host the virtual machine. Compiling the platform-independent code into the native code of the host machine can reduce the execution time of the program and, therefore, increase throughput.

25 [0006] Virtual machines for object-oriented programming languages with dynamic class loading typically load the code of a class when a program resolves a symbolic reference to that class for the first time. The class needs to be initialized

[0007] A class initialization barrier is a sequence of native instructions that calls the virtual machine's runtime to initialize a class if it is not already initialized. Class initialization barriers are included in the implementation of those platform-independent instructions that may result in the very first use of a class (in the case of the Java programming language, there are 4 such instructions: `getstatic`, `putstatic`, `invokestatic`, `new`). The implementation of a platform-independent instruction can come in two flavors: (i) as a sequence of instruction that is part of the implementation of an interpreter of platform-independent instructions, (ii) or as a sequence of instruction generated by a dynamic compiler of platform-independent instructions.

[0008] Because class initialization barriers need only to be executed once per class, it is common practice in the implementation of non-multitasking virtual machines to have recourse to code-rewriting techniques to remove them and the overhead they induce. In other words, a class initialization barrier can simply be